WinHEC 99 White Paper

Windows® Hardware Engineering Conference:
Advancing the Platform

Address Windowing Extensions and Windows 2000 DataCenter Server

Abstract

This article briefly addresses memory space issues in Microsoft® Windows® 2000 DataCenter Server, focusing on the Address Windowing Extensions (AWE) API set, which allows user applications to use up to 64GB of physical non-paged memory in a 32-bit virtual address space on 32-bit platforms with windowed views to this physical memory from within the application's virtual address space.

March 18, 1999

Contents

Disclaimer and Copyright

Introduction

The Microsoft® Windows NT® operating system has always provided applications with a flat 32-bit virtual address space that describes 4 gigabytes (GB) of virtual memory. The address space is usually split so that 2GB of address space is directly accessible to the application, and the other 2GB is only accessible to the Windows 2000 executive software. With Windows NT Server 4.0, Enterprise Edition, and Windows 2000 Advanced Server, 32-bit x86-based systems can provide applications with a 3GB flat virtual address space, with the kernel and executive using only 1GB.

This article briefly addresses memory space issues in Windows 2000 DataCenter, focusing on the Address Windowing Extensions (AWE) API set, which allows user applications to use physical non-paged memory beyond the 32-bit virtual address space and to have window views to this physical memory from within the application's virtual address space. The AWE API also contributes to both system and application performance.

Note: The Microsoft Windows 2000 DataCenter operating system will be available to OEMs, IHVs and ISVs at Beta 3 in Q2 1999. After the release to manufacturing of Windows 2000 Advanced Server, another beta of Windows 2000 DataCenter will be provided to a wider audience.

Address Windowing Extensions API

The following summarizes the Address Windowing Extensions API set:

- **VirtualAlloc** with MEM_PHYSICAL flag
- **AllocateUserPhysicalPages** as described in the Windows Platform SDK
- **MapUserPhysicalPages** as described in the Windows Platform SDK
- **FreeUserPhysicalPages** as described in the Windows Platform SDK

AllocateUserPhysicalPages and **FreeUserPhysicalPages** have a general format similar to the following:

```
(
IN HANDLE  hProcess,            // process within which to allocate memory
IN OUT PULONG_PTR NumberOfPages, // size, in pages, of the physical memory to allocate
OUT PULONG_PTR UserPfnArray     // user address to store the allocated frame numbers in
   );
```

The AWE API set is quite small to ensure that development costs and impacts are low. The AWE API provides native page size support— 4K on 32-bit x86-based processors and 8K on Alpha—meaning that there are no unexpected impacts on performance or application behavior due to unexpected page sizes. Many of the problems that can come from code changes (such as memory leaks and other unexpected behavior) should not occur because of the small API set and the bounds that the operating system places on its use.

Note: This API is supported in all versions of Windows 2000 and does not require Windows 2000 Datacenter Server to operate.

Because of the modest resources required to make the related changes, and because the incremental testing impact is also modest, the adoption of the AWE API is expected to be widespread.

The restrictions related to the AWE API allow an extremely fast map/remap capability to be implemented in the operating system to yield the highest possible performance on a given platform.

Note:

- Information on the AWE API set can be found in the Windows 2000 Beta 3 DDK and on a future version of the Windows Platform SDK.
- **GlobalMemoryStatusEx** is used by the application to determine the physical RAM available for use by the AWE physical memory pool.
- **VirtualAlloc** now has a new flag, MEM_PHYSICAL.

Hardware and Software Requirements

Development Platform

The hardware requirements for developing applications that use the AWE API are similar to those for common current development systems and do not require large investments. In any case, the minimum platform is the same as for Windows 2000 Professional and the required development environment. Check with the system manufacturer for specific chipset and BIOS compliance.

Use of two processors in the development system can make it easier to detect synchronization problems. Also, additional memory will make development and testing more rapid. Although it may be possible to develop on a minimal system, remember that AWE physical memory allocations are not paged, are locked down and are so not available for use by the rest of the system.

Testing Platform

Final testing for correct use of the AWE API and use of memory beyond 32 bits should occur on Windows 2000 DataCenter Server for the platform the application is designed, either the Alpha or x86. It is recommended that testing occur on a system with greater than 4GB of physical memory. Platform-specific testing of an application's correct use of the AWE API may become a requirement for the Microsoft BackOffice® logo for applications.

Device and Driver Requirements for Windows 2000 DataCenter

Microsoft has proposed the following set of requirements for devices and drivers designed for Windows 2000 DataCenter systems that must be met for the system or components to be placed on the Hardware Compatibility List (HCL) or to receive a logo under the Windows Logo Program:

- All SCSI-2, SCSI-3, and Fibre Channel storage adapters and device drivers for Windows 2000 DataCenter must be Dual Address Cycle or 64-bit capable.

- All 100BaseT, Gigabit, ATM [OC3 and higher], FDDI, 16/100 Mb TR, and SAN adapters and device drivers for Windows 2000 DataCenter must be Dual Address Cycle or 64-bit capable.

- Every device and its 64-bit driver must be tested in 8 GB Intel Architecture systems running PAE or in Alpha systems similarly equipped, even if the device has a 32-bit driver and is listed on the general Windows 2000 HCL.

- PCI adapters restricted to 32-bit address space will not be listed on the HCL for Windows 2000 DataCenter, but will be supported if they are not on the primary data path for the DataCenter system. The primary data path is defined as: 1) the highest performance network connection(s) in the system for client-to-server and/or server-to-server communications, or 2) the path(s) on which the majority of storage capacity or the system paging file(s) exist.

  As previously defined in the workstation requirements for the Windows Logo Program, All PCI adapters, including 32-bit PCI adapters, must be able to address the full physical address space on a 64-bit platform. For 32-bit PCI adapters, this means that they must be able to support the Dual

Address Cycle (DAC) command to permit them to transfer 64-bit addresses to the adapter or device (that is, addresses that are above the low 4 GB address space). Adapters that cannot provide this support will not be able to access the full address space on a 64-bit platform.

Additional information is available to members of the PCI Special Interest Group in this area as well as relating to 64-bit PCI adapters and relevant standards. The General Component Protocol Checklist from the PCI SIG includes approximately 45 tests either related to or specifically regarding Dual Address Cycle adapters. See ftp://ftp.pcisig.com/pub/vendors/pcisig/cklst21.zip.

- No ISA cards will be tested, validated, or listed on the HCL or supported by Microsoft for Windows 2000 DataCenter.

- For EIDE, ATA, 10BaseT, 4 MB Token Ring, and so on: these adapters are relatively low-bandwidth and are unlikely to be used for heavy I/ O loads. They will not be included on the HCL for Windows 2000 DataCenter, but may be needed for remote control access or other purposes

For more information about the requirements for the Windows Logo Program, see http://www.microsoft.com/hwdev/winlogo/99logo.htm. See also *Hardware Design Guide Version 2.0 for Microsoft Windows NT Server,* available at http://www.microsoft.com/hwdev/sdg.htm.

Microsoft cannot verify every possible card of the thousands that exist in order to rigorously test the adapters and drivers for Windows 2000 DataCenter Server. Therefore, manufacturers should choose their highest performing, most reliable adapters for this effort.

These requirements are proposed to go into effect when the Windows 2000 DataCenter operating system ships. If you have comments or questions about these requirements, please send e-mail to **hwlogo@microsoft.com**. Please include your name, title, company name, and phone and fax numbers.

API Requirements

The AWE APIs are supported on all Windows 2000 platforms, but it is easy to confuse the APIs with the underlying processor-specific platform memory support. Separate Alpha and x86 binaries are required for applications that support both platforms as the native page size of the platforms is being made visible at the API level. The mechanisms used for greatest performance also impact the level of abstraction the operating system can provide to the developer, regardless of the language used.

This raises the questions of other APIs already in existence for Microsoft Windows 2000, such as the Alpha VLM API. Microsoft has no plans to remove VLM APIs because vendors are coding and shipping VLM-enabled code to customers.

Another pertinent feature is the 4 GB Tuning (4 GT) feature of Windows 2000 Advanced Server on the IA32 platform, which allows 3 GB of memory for user space rather than the usual 2 GB. This will continue to be supported, and IMAGE_FILE_LARGE_ADDRESS_AWARE will continue to work on systems with smaller amounts of physical memory (12 – 16 GB, or less). The transition point is due to kernel virtual memory space considerations. Thus, if the system boots with the **/3GB** entry in Boot.ini, and the system has more than 16 GB of physical memory, the additional physical RAM will not be used by the operating system. Booting without the **/3GB** switch will allow use of all of the physical memory.

Finally, the maximum image (executable file) size remains 512 MB for Windows 2000, regardless of physical memory.

The AWE API is supported in all 64-bit versions of Windows 2000 on all processor platforms. However, because the API invokes low-level operations affected by processor type and version, this impacts the ability of the operating system to emulate the API calls. Consequently, 32-bit applications using AWE

calls will need to be recompiled for 64-bit processors, even though most other Win32-based applications can run in emulation mode on either Intel IA64 or Alpha processors running 64-bit versions of Windows 2000.

Programming Considerations and Restrictions

Protection

Protection is very strict when using the AWE API:

- It is not possible to map the same physical range of memory into two processes. The call will fail.
- It is also not possible to simultaneously map the same physical page from two AWE regions, even in the same process. This is due to limited kernel virtual address considerations.
- Both of the above are due to limited kernel virtual address considerations.

The restriction does require that inter-process communications (sharing techniques such as **MapViewOfFile**) must occur in each process'snon-AWE virtual address space, *not* in the physical memory regions either used as AWE windows or mapped by AWE. However, this should impose no significant performance penalty on well-designed applications that transmit results, rather than raw data to other processes.

A primary reason for the restrictions within the API can be traced to the requirement to provide maximum performance impacts and fact that virtual address space is already largely utilized on heavily loaded systems. There is not enough kernel virtual memory space on a 32-bit platform to manage the sharing of 36-bit physical addresses.

No Executables or Buffers within an AWE Window

The AWE API does not permit executable code (.exe, .dll, .sys, and so on) to execute from within an AWE window region in the process's virtual memory or in the mapped physical memory pool the process is utilizing.

A similar restriction is that AWE window address ranges and memory pools cannot be used as data buffers for graphics or video calls.

Applications using the AWE API set must have the Lock Pages in Memory privilege in order to allocate physical memory. For samples and more information, see the AWE API information in the Microsoft Platform SDK.

Finally, it is not possible to free only part of the AWE memory allocation made by a thread or process. Each AWE memory allocation must be freed as a unit.

AWE Application Considerations

The application developer should keep in mind several factors for setting AWE window sizes and the size of the physical memory pool allocated.

- **How much data locality does the application exhibit?**
  Some applications, such as databases, have little locality after operating for long periods, whereas scientific and engineering applications may have considerable locality.

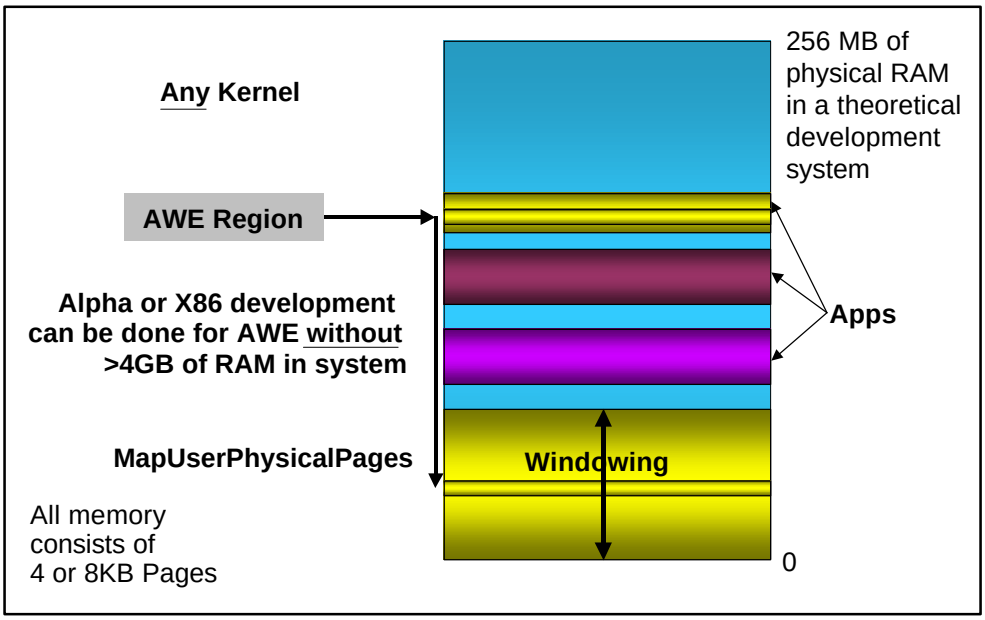
- **What is the application's performance behavior in response to more memory versus processor or input/output performance?**
  This is another way of saying that by alleviating one bottleneck (physical memory space in this case) the limiting performance factor will become some other part of the system—network or

storage, for example. Other factors are the costs associated with the AWE calls and functions. AWE allocations and frees are relatively slow while the remaps are extremely fast.

- **What is available physical memory in the system, considering memory is not fixed, even on the same system over time?**
  Different customers will have differing configurations, and the configuration may change as memory is added or removed. This pool of physical memory is a real—and limited—resource, and inadvertently locking all of it will negatively impact all other functions of the system. Consequently, developers of AWE applications must be careful not to take so much physical memory that they cause other applications to page excessively or prevent creation of new processes or threads due to lack of resources. Use the **GlobalMemoryStatusEx** function to check the physical memory size.

- **How can the developer give the administrator the ability to control the total RAM used by the application to prevent 'hogging'?**
  Given the information cited here, the application should be able to calculate an optimum AWE window size for that installation and instance of the application for a given system.

Figure 1. Developing AWE Applications



Administration and Management

Task Manager

Once the system is operational, the easiest first choice for examining the system memory 'load' is Task Manager. This is accessible to the administrator in a number of ways and quickly indicates the total physical memory, the 'free' memory, and the amount used by the cache. Task Manager is capable of displaying the total physical memory in the system and is not limited to 4 GB. Notice that essentially Task Manager returns the same type of information as does a call to **GlobalMemoryStatusEx**.

Performance Monitor

Performance Monitor (PerfMon) is the next tool that can be used to examine the state and usage of system memory. PerfMon provides a greater level of detail and is also useful for examining the interrelationships

that may exist in various parts of the operating system with regards to memory use. For systems with large amounts of memory that are hosting applications using the AWE API set, two PerfMon counters are provided:

- The Per Process Physical Allocations counter provides information about specific processes and their use of the AWE API for allocations. Specifically, this reports information related to the **AllocateUserPhysicalPages** call. This counter can have an instance for each process/thread.
- The related counter, System Total Physical Allocations, provides information on the total of all physical memory allocations the system is supporting for all applications or processes using AWE functions. Given this information, as well as the PerfMon object/counter named Memory/Available Bytes, an administrator can judge the current memory load on the system and ensure that any applications loaded do not cause memory pressure and excessive paging.

Job Object

It is important to note that although the Job Object API and management tools can control virtual memory usage for any application, whether or not it is using the AWE API set, there is no control over the physical memory pool that is available to a application from an AWE window region. The AWE window region appears in the process's virtual address space, but the physical pool it accesses is not visible through the use of the Job Object API.

Uses and Advantages of AWE and Large Memory

A number of application classes can benefit from large memory: databases (both relational and object oriented), enterprise resource planning, decision support, scientific and engineering applications such as computational fluid dynamics or finite element analysis, and financial analysis applications—all of which manipulate large data sets and/or calculate from large data sets. In all these cases, physical memory is being used to cache references to, and compensate for, the very much larger latencies inherent in mass storage devices.

Windows 2000 DataCenter Server and Large Memory

Windows 2000 DataCenter Server provides both native Physical Address Extension [PAE] support on the Intel IA32 processor for making available 64 GB, and extended addressing support on the Alpha, for making available 32 GB. This affords a number of benefits:

- All the physical memory in the system can be treated as general-purpose memory. The operating system is able to use this memory for caching and virtual memory management without major changes.
- Applications can also use this memory without changes, the most noticeable difference being that more applications can be running before paging occurs, as there is more memory for each process and the associated virtual address space of each. This also means that more processes can be resident. Because applications not needing larger physical memory do not need to be modified or use the AWE API, virtually all applications can benefit.
- Those applications that do require more physical memory than is provided by the 4 GB virtual address space can use the AWE API set to map from a designated window into the physical memory region allocated for that window.
- Direct I/O is supported for the entire physical address space if the network and storage devices and drivers support dual address cycle and 64 bit addressing.
- The kernel virtual memory space organization for PAE-X86 and Alpha is unchanged from the standard Windows 2000 kernel, providing an easier migration path for the device drivers. For example, the maximum non-paged pool remains at 256 MB. Similarly, the current configuration

settings and their effects are identical, such as the use of the **/3GB** switch in the Boot.ini file, which allocates 3 GB of virtual address space to an application that uses IMAGE_FILE_LARGE_ADDRESS_AWARE in the process header.

Advantages of the AWE API Set for Using Large Memory

Several advantages are gained in using the AWE API set and Windows 2000, in comparison to other solutions.

- The memory allocations are both 'finer grained' and faster than that for a complete process space creation or 'fork.'
- Mapping is exceptionally fast due to more direct use of the underlying hardware platform's native capabilities.
- The API set is quite small and easily implemented.
- Applications needing windowing and using the AWE API gain the greatest possible benefit because the remap operations are very optimized.
- Use of the API set is not required, so applications not requiring more physical memory for optimal performance are not affected and run transparently.

Figure 2. Consolidation Server

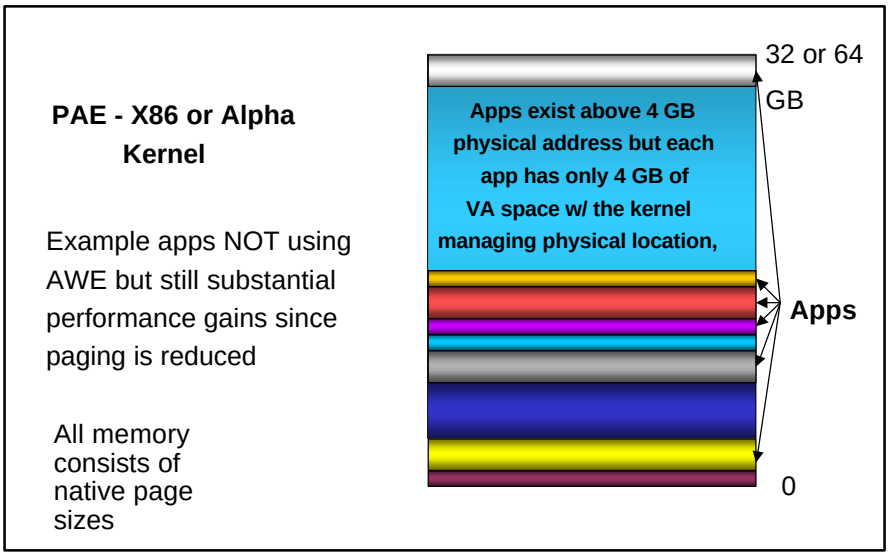
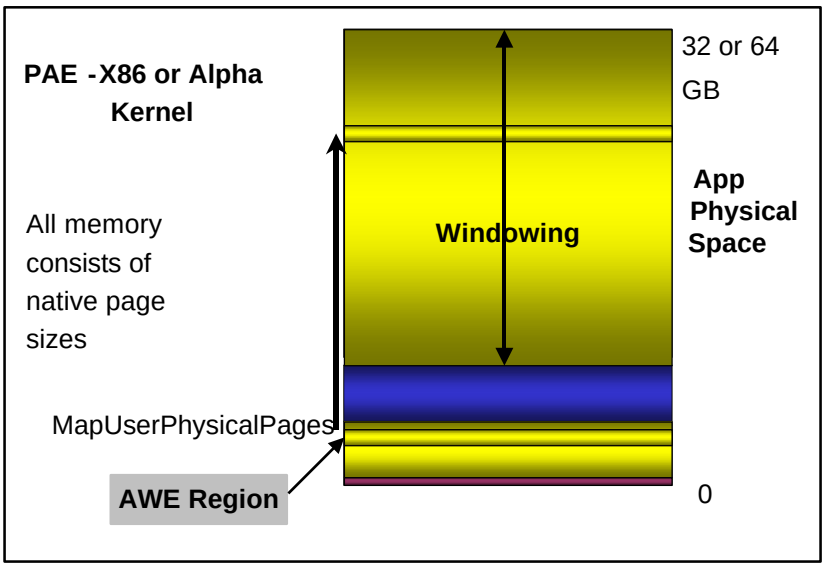


Figure 3. Single AWE Application



Large Memory and System Impact

The impact of large memory on the rest of Windows 2000 DataCenter operations is for the most part completely transparent with some exceptions:

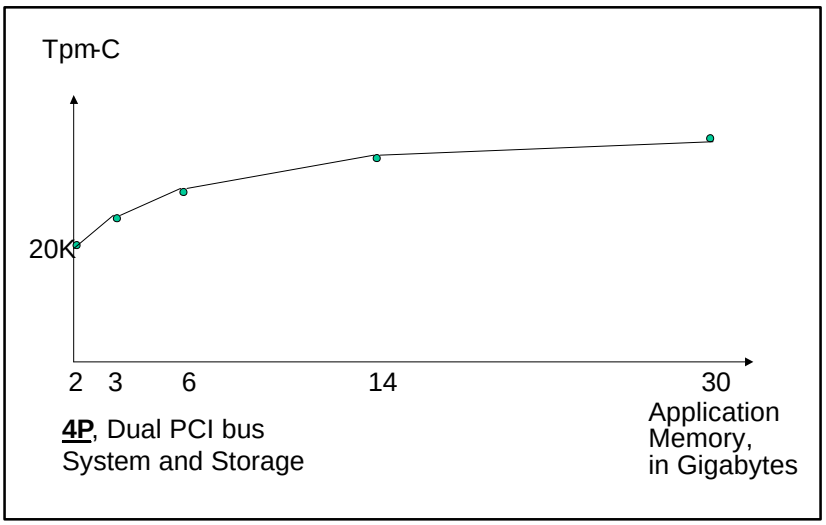
- Crashdump options must be set for 'Kernel Information Only,' which captures the critical information required for analysis.
- Testing and analysis tools such as those in the Windows 2000 DDK and Microsoft Hardware Compatibility Tests (HCTs), the Driver Verifier facility, the debuggers, and so on all work without any changes required.
- Services and software devices (IFS device drivers, and so on) that are included in the system continue to operate without change.

Performance

The primary benefit and advantage of the AWE API set and large memory is, of course, application and operating system performance. As mentioned earlier, memory alone is not a panacea for all performance requirements. Diminishing relative returns can be expected if the only modification to the platform is additional memory. This is why it is important to achieve a performance balance among processors, memory, and I/O. Otherwise, performance tapers off to the point where additional memory provides no increase in functionality. The limiting factor becomes some other part of the system.

This is illustrated in Figure 3 by two hypothetical example systems running an OLTP workload. Note that there are substantial and worthwhile performance increases for a 4-processor system when an application can have access to more memory, but only to a certain point. The first example shows a hypothetical application's performance using 2 GB of physical memory, increasing to 3 GB and reaching an 'inflection point' at perhaps 6 GB. Further increasing memory to 14 GB has a negligible impact on performance. Essentially, some other part of the system is 'saturated' and additional memory alone does not affect the performance.

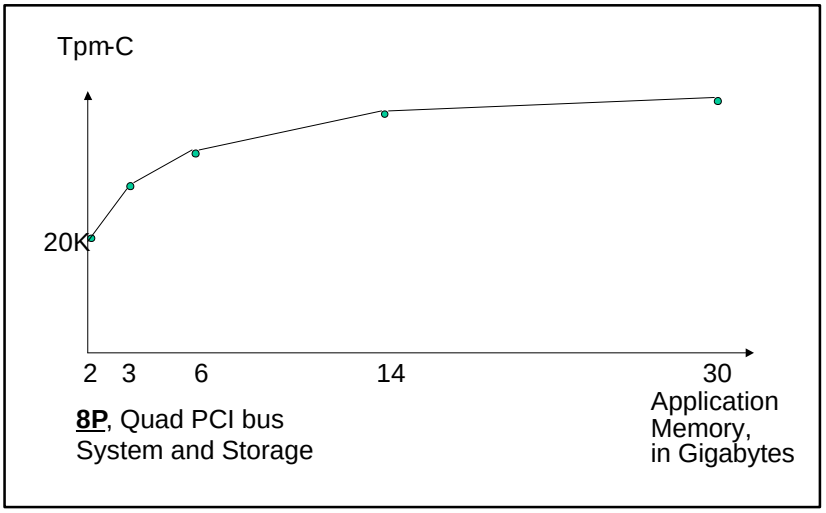
Figure 4. Expected Performance Impact of Additional Memory



The second example shows a more 'balanced' system consisting of more processors and I/O, in this case 8 CPUs and 4 PCI buses. In this example, there are two effects to notice:

- Increase in performance for each change in memory size is greater at the lower end of the memory scale. This would indicate the system was memory bound, and that the processors and I/O were not fully utilized.
- The 'inflection point' moves to a larger amount of memory before the rate of increase becomes low.

Figure 5. Balanced System Performance



As mentioned earlier, given that the Windows 2000 virtual memory manager will use all available memory, the possibility that large AWE physical memory allocations can be by made by an application and that these allocations are not paged, it is possible for excessive paging to occur for other applications already resident in memory.

Finally, it is important for best performance that the I/O hardware be either DAC or 64-bit capable, and the associated device drivers must be capable of addressing 64-bit physical memory. If this is not the case for an I/O intensive device such as a network adapter or host bus adapter, then double-buffering of I/O occurs. Although this is 'transparent' in that the operating system manages these double buffer operations,

the performance impact is large for high performance I/O devices such as 100BaseT, 1000BaseT, FDDI or ATM network adapters and for storage adapters such as those for SCSI-2, SCSI-3 or Fibre Channel. To gauge the impact, the present example of worst performance is ISA, where devices can only physically address the first 16 MB due to 24-bit addressing limitations.

The impact grows as the physical memory in the system increases. For example, on a system with 4 GB of physical memory, if the system has no ISA, no double buffering occurs, and there is no impact. On a system with 8 GB of physical memory, it can be expected that more than one half of all I/O operations would occur from addresses above 4 GB. A system with 16 GB of physical memory would have more than three-quarters of all I/O operations originating at addresses beyond 4 GB.